Lecture 7

Perturbation

Computational geometers tend to formulate their algorithms for inputs in general position. What is an input in general position? General position is always defined with respect to a set of predicates. A set of points is in general position with respect to the orientation predicate if no three points are collinear. It is in general position with respect to the side-of-circle predicate if no four points are co-circular. It is general position with respect to the orientation predicate and the side-of-circle predicate if no three points are collinear and no four points are co-circular. Generally, if $f_1, \ldots, f_k$ are functions of geometric objects, then a set of objects is in general position with respect to these functions, if all function evaluations for objects in the set yield nonzero.

Geometric algorithms branch on the outcome geometric predicates. In general, the branches are three-way branches: positive sign, negative sign, and zero. If the input is in general position, the zero branch is never taken. This simplifies the algorithm. We have already seen several examples to this effect. In the convex hull algorithm, we had to distinguish between visibility and weak visibility and we had to cope with inputs that are contained in a lower dimensional subspace. In the Delaunay triangulation algorithm, we had to cope with co-circular points and with inputs that are contained in a lower dimensional subspace.

So the general position assumption simplifies the life of an algorithm designer. However, at the cost of the programmer. A program has to cope with all inputs and so has to deal with degenerate inputs. What can a programmer do? There are essentially two approaches:

- Redesign the algorithm so that it handles degenerate inputs.
- Use perturbation to bring the input into general position.

Whenever we discuss an algorithm in this book, we follow the first approach. We make sure that the algorithms works for all inputs. In this lecture and the next, we study perturbation techniques. *The idea is to solve the problem not on the given input, but on a nearby input. The nearby input is obtained by perturbing the given input. The perturbed input will then be in general position and, since it is near the original input, the result for the perturbed input will hopefully still be useful.* This hope has to be substantiated in any application of the perturbation technique. We cannot make general claims with respect to this hope. We give a positive and a negative example. If the input objects are derived from some physical measurement, then a perturbation within the precision of the measuring device should be acceptable. On the other hand, for an algorithm whose task is to decide whether the input is in general position, perturbation makes no sense.

Exercise 0.1: Go through the examples in the first lecture. For which of them is perturbation a reasonable technique? Discuss two additional examples of your own choice. ♢

Perturbation comes in two flavors: symbolic and numerical. In symbolic perturbation, one perturbs inputs by infinitesimal amounts, and in numerical perturbation, one actually changes the coordinates. (REWRITE).

7.1 Symbolic Perturbation

It is convenient to summarize the input into a single vector $x \in \mathbb{R}^N$. For example, if the input is n points in the plane, we would set $N = 2n$ and pack all $2n$ coordinates into a single vector. A test function is then simply a function $f : \mathbb{R}^N \mapsto \mathbb{R}$. Let F be a collection of test functions. For example, if an algorithm uses the geometric predicates lex-compare, orientation, and side-of-circle for n points in the plane, F contains $\binom{n}{2}$ test functions corresponding to lex-compare (one for each pair of distinct points), $\binom{n}{3}$ test functions corresponding to orientation, and $\binom{n}{4}$ test functions corresponding to side-of-circle.

DEFINITION 1. *Let $f : \mathbb{R}^N \mapsto \mathbb{R}$ be a test function and $\sigma = f^{-1}(0)$ be its zero set. We call f* well-behaved *if every straight line ℓ is either contained in σ or every bounded segment of ℓ intersects σ in finitely many points.*

Many functions are well-behaved, e.g., all polynomials and all rational functions. In particular, for any geometric test used in this book, the underlying function is well-behaved.

THEOREM 1. *Let F be a collection of well-behaved continuous functions and let $a \in \mathbb{R}^N$ be a vector that is in general position with respect to F, i.e., $f(a) \neq 0$ for all $f \in F$. Then for any $f \in F$ and any $q \in \mathbb{R}^N$*

$$\overline{f}(q) := \lim_{\varepsilon \to 0+} \operatorname{sign} f(q + \varepsilon(a - q))$$

exists and is non-zero. Moreover, if $f(q) \neq 0$, $\overline{f}(q) = \operatorname{sign} f(q)$.

Proof. The function $\varepsilon \mapsto q + \varepsilon(a - q)$ defines a line ℓ passing through q and a. Since $f(a) \neq 0$, ℓ is not contained in σ and hence the segment $\overline{qa}$ intersects σ only finitely often. Thus there is an $\varepsilon_0 > 0$ such that $f(q + \varepsilon(a - q)) \neq 0$ for $0 < \varepsilon < \varepsilon_0$. Since f is continuous, $\operatorname{sign} f(q + \varepsilon(a - q))$ is constant for $0 < \varepsilon < \varepsilon_0$. Thus $\overline{f}(q)$ exists and is non-zero.

Assume next that $f(q) \neq 0$. Since f is continuous, there is an $\varepsilon_0 > 0$ such that $f(q + \varepsilon(a - q)) \neq 0$ for $0 \leq \varepsilon < \varepsilon_0$. Again by continuity, $\operatorname{sign} f(q) = \overline{f}(q)$. □

COROLLARY 2. *Consider any algorithm that branches only on the sign of a function f from a class F of well-behaved continuous functions applied to the input $q \in \mathbb{R}^N$. Also assume that $a \in \mathbb{R}^N$ that is non-degenerate for all $f \in F$. Branching on $\overline{f}(q)$ instead of on $\operatorname{sign} f(q)$ has the following effect:*

- *The zero branch is never taken, and*
- *If q is in general position, the computation does not change.*

Proof. This follows immediately from Theorem 1. Since $\overline{f}(q) \neq 0$ for all q, the zero branch is never taken, and since $\overline{f}(q) = \operatorname{sign} f(q)$ whenever $f(q) \neq 0$, the computation does not change for an input in general position. □

The corollary may be paraphrased as *if you know just one input in general position, any input can be perturbed into general position*. We still need to address two questions. How do we find inputs in general position and how can we compute $\overline{f}(q)$? We address both questions first for the orientation predicate of n points in the plane.

LEMMA 3. *The points $a_i = (i, i^2)$, $1 \leq i \leq n$, are in general position with respect to the orientation predicate.*

Proof. Lines intersect the parabola $y = x^2$ in at most two points. Thus no three a_i are collinear. □

We next discuss how to evaluate the orientation predicate. Assume our inputs are the points q_i, $1 \leq i \leq n$. We replace q_i by $q_i + \varepsilon(a_i - q_i)$. For three distinct points q_i, q_j, and q_k, we then have:

$$\overline{\mathsf{Orientation}}(q_i,q_j,q_k) = \lim_{\varepsilon\to 0+} \operatorname{sign} \begin{vmatrix} 1 & (1-\varepsilon)x(q_i)+\varepsilon i & (1-\varepsilon)y(q_i)+\varepsilon i^2 \\ 1 & (1-\varepsilon)x(q_j)+\varepsilon j & (1-\varepsilon)y(q_j)+\varepsilon j^2 \\ 1 & (1-\varepsilon)x(q_k)+\varepsilon k & (1-\varepsilon)y(q_k)+\varepsilon k^2 \end{vmatrix}$$

Expansion and collecting terms according to powers of ε yields

$$= \mathsf{Orientation}(q_i,q_j,q_k) + \lim_{\varepsilon\to 0+} \operatorname{sign} \left(\varepsilon P(q_i,q_j,q_k) + \varepsilon^2 \begin{vmatrix} 1 & i & i^2 \\ 1 & j & j^2 \\ 1 & k & k^2 \end{vmatrix} \right),$$

where $P(q_i,q_j,q_k,i,j,k)$ is a polynomial. Thus

$$\overline{\mathsf{Orientation}}(q_i,q_j,q_k) = \begin{cases} \mathsf{Orientation}(q_i,q_j,q_k) & \text{if } \mathsf{Orientation}(q_i,q_j,q_k) \neq 0 \\ \operatorname{sign}(P(q_i,q_j,q_k)) & \text{if } \mathsf{Orientation}(q_i,q_j,q_k) = 0 \text{ and } P(q_i,q_j,q_k) \neq 0 \\ \mathsf{Orientation}(a_i,a_j,a_k) & \text{if } \mathsf{Orientation}(q_i,q_j,q_k) = 0 = P(q_i,q_j,q_k) \end{cases}$$

We next address the equations more generally. We exhibit inputs in general position for the set of test functions introduced in the introductory paragraph. We do so for arbitrary dimension d and not only for the plane. We consider n points chosen from the positive branch (i.e., $t > 0$) of the moment curve $t \mapsto (t, t^2, \ldots, t^d)$. No two points on this curve agree in any coordinate. No $d+1$ points lie in a common hyperplane. Consider the equation $a_0 + \sum_{1\leq i\leq d} a_i x_i$ of any hyperplane. Plugging $x = (t, t^2, \ldots, t^d)$ into this equation gives a polynomial of degree d in t. We conclude that the hyperplane intersects the moment curve in at most d points. Finally, the positive branch of the moment curve intersects no sphere in $d+2$ or more points. Let $\sum_{1\leq i\leq d}(x_i - c_i)^2 - r^2 = 0$ be the equation of a sphere. Plugging $x = (t, t^2, \ldots, t^d)$ into this equation gives the following polynomial in t:

$$\sum_{1\leq i\leq d} (t^i - c_i)^2 - r^2.$$

Descartes rule of signs (Theorem **??**) states that the number of positive roots of a polynomial is bounded by the number of sign changes in its coefficient sequence. The polynomial above can have at most $d+1$ sign changes since the coefficients of the powers t^j with $j > d$ are nonnegative (any such coefficient is either zero or one).

We first show how to compute $\overline{f}(q)$ for polynomials f. We use q_1 to q_N to denote the coordinates of $\mathbb{R}^N$ and assume that $f(q_1, \ldots, q_n)$ is a polynomial of total degree d. Then.

$$f(q + \varepsilon(a-q)) = f(q_1 + \varepsilon(a_1 - q_1), \ldots, q_N + \varepsilon(a_N - q_N)) = \sum_{0\leq i\leq d} p_i(q_1, \ldots, q_n)\varepsilon^i,$$

where the p_i are polynomials of total degree at most d. We claim

$$\overline{f}(q) = \text{sign} p_i(q) \quad \text{where } i = \min\{j \mid p_j(q) \neq 0\}.$$

We know from Theorem **??** that the sign of $f(q+\varepsilon(a-q))$ is constant and nonzero for sufficiently small ε. Therefore at least one $p_j(q)$ must be non-zero. Let i be minimal with $p_i(q) \neq 0$. Then

$$f(q+\varepsilon(a-q)) = p_i(q)\varepsilon^i \left(1 + \sum_{j>i} \frac{p_j(q)}{p_i(q)} \varepsilon^{j-i}\right).$$

Let $M = \max \left|p_j(q)/p_i(q)\right|$. Then $\left|\sum_{j>i} p_j(q)/p_i(q)\varepsilon^{j-i}\right| \leq M/(1-\varepsilon) < 1/2$ for sufficiently small ε.

7.2 Numerical Perturbation

[[the following is copied from Funke/Klein/Mehlhorn/Schmitt. It needs to rewritten so that it fits better.]]

7.3 Some Words of Caution

Perturbation is not a cure-all. It removes burden from the algorithm designer and implementer. However, it has two drawbacks.

The running time of an algorithm may increase as a result of perturbation. We give two examples. Assume we are given n line segments passing through the origin. We will see in Section **??** that we can compute their arrangement in time $O(n \log n)$. However, perturbing the line segments into general position (no three intersect in a point) will generate an arrangement with $\Theta(n^2)$ intersection points. The second example is even more extreme. Assume we are given n identical points in $\mathbb{R}^d$. Any sensible convex hull algorithm should be able to handle this input in linear time. However, the perturbation scheme of Section 7.1 moves the n points onto the d-dimensional moment curve. The resulting hull will have $\Omega n^{\lfloor d/2 \rfloor}$ facets and hence any algorithm will need time $\Omega n^{\lfloor d/2 \rfloor}$ for computing the hull of the perturbed points.

Exercise 0.2: Prove bound for points on the moment curve. ♢

The second drawback is that we solve the problem on a perturbed input and not on the original input. The output for the perturbed input may tell us little about the output for the original input.

The symbolic scheme has another drawback. It requires exact computation.

Neither approach to perturbation will apply if some test function is identically zero. For example, if one tests whether a point p lies on a line involving p as one of the defining points, the outcome will be "on line" no matter who one perturbs the input. The reader may think that test functions that are identically zero can only arise as a consequence of stupid programming. However, they can also arise because the algorithm designer misses a theorem, see Figure 7.1.

7.4 Notes

[3] introduced symbolic perturbation and applied it to the orientation predicate. [5, 7, 4, 8] extended and simplified the technique. Our presentation follows [7]. An implementation of the scheme is available in CGAL [2]. CITATION IS INCOMPLETE.

Section **??** is based on [6].

Section 7.3 is based on [1].

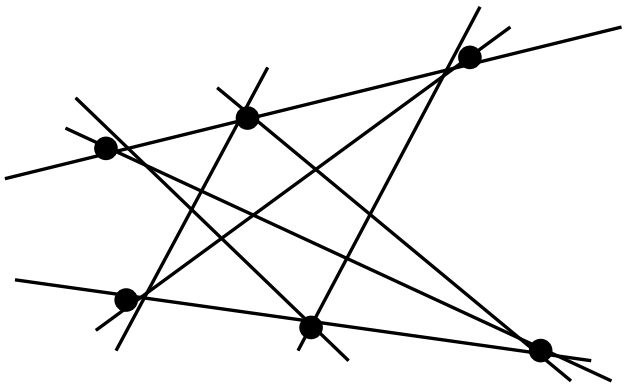

Figure 7.1: p_1, p_2, p_3 are three arbitrary points on a line ℓ_1 and q_1, q_2, q_3 are three arbitrary points on a line ℓ_2. For $1 \leq i \leq 3$ let $\{j,k\} = \{1,2,3\} \setminus i$ and let r_i be the intersection of $\ell(p_j,q_k)$ and $\ell(p_k,q_j)$. Pappus (ca. 300 AD) proved that r_1, r_2 and r_3 are collinear. So perturbing the input will not help.

7.5 Proposed Contents

discuss SoS by Edelsbrunner and Muecke, Seidel

discuss controlled perturbation. This can be based on the SODA article by Funke/Klein/Mehlhorn/Schmitt. Reference to Devillers/Preparata.

also do conceptual perturbation: walk through a triangulation. to get the code right. This is discussed in the LEDAbook and also in my 2000 course notes.

Bibliography

[1] C. Burnikel, K. Mehlhorn, and S. Schirra. On Degeneracy in Geometric Computations. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94)*, pages 16–23, 1994.

[2] J. Comes and M. Ziegelmann. An easy to use implementation of linear perturbations within CGAL. In *Algorithm Engineering*, pages 169–182, 1999.

[3] H. Edelsbrunner and E. Mücke. Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms. *ACM Transactions on Graphics*, 9(1):66–104, Jan. 1990.

[4] I. Emiris, J. Canny, and R. Seidel. Efficient perturbations for handling geometric degeneracies. *Algorithmica*, 19:219–242, 1997.

[5] I. Z. Emiris and J. F. Canny. A general approach to removing degeneracies. *SIAM Journal on Computing*, 24(3):650–664, June 1995.

[6] S. Funke, C. Klein, K. Mehlhorn, and S. Schmitt. Controlled Perturbation for Delaunay Triangulations. SODA, pages 1047–1056, 2005.

[7] R. Seidel. The nature and meaning of perturbations in geometric computing. *Discrete & Computational Geometry*, 19(1):1–17, 1998.

[8] C.-K. Yap. Geometric consistency theorem for a symbolic perturbation scheme. *J. Comput. Syst. Sci.*, 40(1):2–18, 1990.